GETPASS

Vulnerable to internal buffer overflows

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 3863 bytes

Attack Category	• Malicious Input
Vulnerability Category	• Buffer Overflow • Input source (not really attack) • Unconditional
Software Context	• String Management
Location	
Description	Some versions of getpass() allow overflow of an internal buffer. The getpass function is designed to accept a password from the console, which is a null-terminated string. The echo is off so it will not appear on the screen. It can lead to a buffer overflow problem, but that is very implementation dependent. In some implementations of the function, there is a maximum length defined for the password, and in other implementation, the password can be of arbitrary length.
APIs	<table><tr><th>Function Name</th><th>Comments</th></tr><tr><td>getpass</td><td></td></tr></table>
Method of Attack	Password entry is controlled by the user, who could potentially mount a buffer overflow attack.
Exception Criteria	
Solutions	<table><tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr><tr><td>Whenever password must be read from console.</td><td>Consult bug/ security notices relevant to the particular implementation of getpass(). Consider use of readpassphrase()</td><td>Effective if implementation is sound, or if one can substitute a function with a sound implementation.</td></tr></table>

[1] http://buildsecurityin.us-cert.gov/bsi/about_us/authors/35-BSI.html (Barnum, Sean)

<table>
<tr><td></td><td><table><tr><td></td><td>on platforms that support it.</td><td></td></tr></table></td></tr>
<tr><td><b>Signature Details</b></td><td>char *getpass(const char* prompt);</td></tr>
<tr><td><b>Examples of Incorrect Code</b></td><td><code>char *passwd = getpass("Enter password:");</code></td></tr>
<tr><td><b>Examples of Corrected Code</b></td><td><code>char passwd[passwdBuffSize];
if (!readpassphrase("Enter password:", passwd, passwdBuffSize, 0))
{ handleError(); }</code></td></tr>
<tr><td><b>Source Reference</b></td><td>• Viega, John & McGraw, Gary. <i>Building Secure Software: How to Avoid Security Problems the Right Way</i>. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X , p. 148.</td></tr>
<tr><td><b>Recommended Resource</b></td><td></td></tr>
<tr><td><b>Discriminant Set</b></td><td><table><tr><td><b>Operating System</b></td><td>• Windows</td></tr><tr><td><b>Languages</b></td><td>• C
• C++</td></tr></table></td></tr>
</table>

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:copyright@cigital.com